CREATE TABLE

ALTER TABLE

DROP TABLE

INSERT INTO

Retrieving Data

Data can be retrieved from SQL tables using the SELECT statement. SELECT is a declarative instruction which allows the user to specify the criteria for matching data in the database without specifying exactly how that data is retrieved. The SQL Server will decide the optimum way to fetch the data.

Syntax:

<SELECT clause> <From clause> [Where clause] [Group By clause] [Having clause] [Order By clause]

(Square brackets = optional)

Retrieving data from a single table

SELECT <columns> FROM <Table Name>;

Columns is a comma separated list of columns required, or \* if all columns are needed

e.g. list all details of all users:

SELECT \* FROM User;

List all details of users from county Wexford:

SELECT \* FROM User WHERE County=”Wexford”;

Retrieving data from multiple tables

Most business application transactions require data to be retrieved from multiple tables simultaneously. The data might be held in different tables because the data was normalised. In such case the tables are related using a foreign key.

When retrieving data from such tables the foreign key value can be used to 'Join' the two tables back together again, e.g.

User

|  |  |
| --- | --- |
| UserID | User |
| 10 | John |

Loan

|  |  |  |
| --- | --- | --- |
| LoanID (PK) | Name | UserID (FK) |
| 100 | Book 1 | 10 |
| 200 | Book 2 | 10 |

Loan & User ('joined' by program in memory as if one big record)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LoanID | Name | UserID | UserID | User |
| 100 | Book 1 | 10 | 10 | John |
| 200 | Book 2 | 10 | 10 | John |

Tables can be joined by listing multiple tables in the FROM clause and specifying a foreign key match in the WHERE clause e.g.

List the loan due date and the user for every loan:

SELECT name,dueDate FROM User,Loan WHERE User.USERID = Loan.LOANED\_TO\_USER;

CRUD Operations

Create – INSERT INTO

Read – SELECT

Update – UPDATE

Delete - DELETE

We need to determine what database operations are needed to accomplish each business function (use case). This would already have been done as part of the specification and transaction analysis activities.­

Example – The 'borrow book' use case:

|  |  |  |
| --- | --- | --- |
| Use Case Activities (Outside World) | Database Operations (System) | SQL (System) |
| User presents book to librarian | n/a | n/a |
| Librarian requests user ID card | n/a | n/a |
| Librarian retrieves book details | READ Book table for records matching Book ISBN | SELECT \* FROM BOOK  WHERE ISBN = "<input ISBN>" |
| Librarian enters user ID against book | 1) READ User table for record matching User ID (ensure it exists)  2) CREATE A NEW LOAN record for Book ISBN and User ID | SELECT \* FROM USER WHERE USER\_ID = "<Input User ID>"  INSERT INTO LOAN  VALUES(ISBN, USER\_ID, dueDate,....) |
| User leaves | n/a | n/a |

Updating rows in a table

The SQL Update statement is used to modify the fields of existing table rows. The syntax is:

UPDATE <table> UPDATE USER

Set <field> = <value> e.g. SET addr1 = '1 Main Street'

WHERE <condition>; WHERE user\_id = 123;

This changes User no. 123's address to 1 Main Street. More than one field can be updated using multiple SET commands.

UPDATE Book

SET section = 'B6'

WHERE section = 'B4';

This moves all books currently in section B4 to B6

Deleting rows form a table

To remove rows from a table use the SQL DELETE statement:

DELETE

FROM <table>

WHERE <condition>;

To delete the record for user 456:

DELETE

FROM User

WHERE user\_id = 456;

If there is no condition after WHERE, the whole table will be removed. To delete the actual table definition from the database use the DROP command: DROP TABLE User;

Example – The 'Return Book' use case

Method 1 – Loan data is preserved for future reporting purposes and loan record is marked as 'returned':

|  |  |  |
| --- | --- | --- |
| Use Case Activities (Outside World) | Database Operations (System) | SQL (System) |
| User presents book to librarian | n/a | n/a |
| Librarian requests user ID card | n/a | n/a |
| Librarian retrieves Book record | READ Book record and with it the associated loan ID foreign key | SELECT \* FROM Book  WHERE ISBN = <input ISBN> |
| Librarian updates loan to have been returned | UPDATE the loan record to be 'returned' | UPDATE Loan  SET return\_date = CURRENT\_DATE  WHERE loan\_id = Book.loan (foreign key from Book) |
| User leaves | n/a | n/a |

Method 2 – Delete the loan record and set the associated foreign key field in Book to NULL. Assume that the Book – Loan foreign key has its ON DELETE option configured to 'Set Null'. This will update the Book foreign key field to NULL automatically when we delete the Loan.

|  |  |  |
| --- | --- | --- |
| Use Case Activities (Outside World) | Database Operations (System) | SQL (System) |
| User presents book to librarian | n/a | n/a |
| Librarian requests user ID card | n/a | n/a |
| Librarian retrieves loan details | READ Loan table for records matching Book ISBN | SELECT \* FROM Loan  WHERE ISBN = "<input ISBN>" |
| Librarian removes loan from table using ISBN and USER\_ID | DELETE the Loan record for Book ISBN and User ID | DELETE FROM Loan WHERE LOAN\_ID = Book.LOAN\_FK |
| User leaves | n/a | n/a |

More commands (from sql\_intro.pdf)

List all counties the library users are from:

Input Output

SELECT county Waterford

FROM User; Waterford

Kilkenny

(county is a field, User is the table being searched) Kilkenny etc.

How many Library users are from a specific county e.g. Wexford?

SELECT COUNT(\*)

FROM User

WHERE county = 'Wexford';

Group Users and group them by their county:

SELECT County, COUNT(\*) Waterford 120

FROM User Kilkenny 98

GROUP BY County; Wexford 74

SQL Subqueries

Normally SELECT statements use static conditions in the WHERE clause e.g.

SELECT title, author FROM Book WHERE category IN ['Crime, 'Horror']

Sometimes we may need to find books belonging to a changing/dynamic set of categories. We could store each category required as a set of records in a table e.g.

Table Category\_Required

Horror

Crime

Science

History

Travel

We might add or subtract records from this table before performing the query. To use dynamically-defined conditions in SELECT statements, a subquery can be used. The previous SELECT can be rewritten to use a subquery as follows:

SELECT title, author

FROM Book

WHERE category IN

[SELECT category\_need FROM category\_required]; (this is the subquery)

Subqueries can be used anywhere a scalar (single value) or a set of values are used e.g. in WHERE conditions, after IN, BETWEEN, etc.

Example: List all users with total\_fines above the average

SELECT name

FROM User

WHERE total\_fines >

[SELECT AVERAGE(total\_fines) FROM User]; (AVERAGE is a built-in SQL function)

Database Views

Relational database tables are physically stored on disk. Views on the other hand are like dynamically created tables that are not stored on disk. The fields in these virtual tables are created/populated from underlying real physical tables and they are created dynamically as needed e.g. in SELECT statements

Create a view containing all users from Wexford:

CREATE VIEW Wex\_Users AS

SELECT \* FROM User WHERE County = 'Wexford';